(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 045 307 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
18.10.2000 Bulletin 2000/42

(51) Int. Cl.^{7}: G06F 12/08

(21) Application number: 00106930.1

(22) Date of filing: 31.03.2000

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 16.04.1999 US 293278

(71) Applicant:
Infineon Technologies North America Corp.
San Jose, CA 95112-6000 (US)

(72) Inventors:
- Gautho, Manuel O.
San Jose, CA 95126 (US)
- Afsar, Muhammad
San Diego, CA 92129 (US)

(74) Representative:
Patentanwälte
Westphal, Mussgnug & Partner
Waldstrasse 33
78048 Villingen-Schwenningen (DE)

(54) Dynamic reconfiguration of a micro-controller cache memory

(57) A configurable cache memory (314) for improving the performance of a central processing unit (CPU) (304) in the execution of a program is described. The configurable cache memory (314) provides scratch pad RAM based upon the particular requirements of the program being executed by the CPU (304). The CPU (304) provides configuration data based upon the program that is used by the configurable cache memory (314) in reallocating its memory space accordingly.

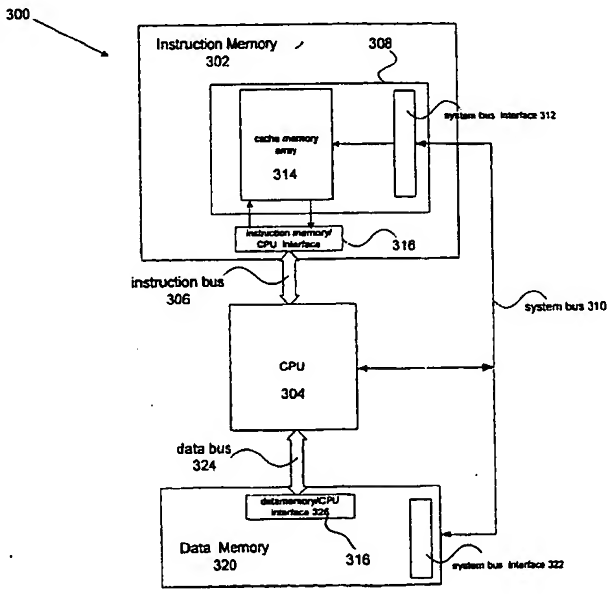


Figure 3

EP 1 045 307 A2

Description

FIELD OF THE INVENTION:

[0001] The present invention pertains to computing systems. More specifically, the present invention relates to dynamically reconfiguring the cache memory of micro-controllers, and the like.

BACKGROUND OF THE INVENTION:

[0002] Figure 1 shows a conventional computer system 100. The computer system 100 includes a central processing unit 102, or CPU, that operates in accordance with a pre-determined program or set of instructions stored within an associated program memory 104. Typically, instruction processing generally includes fetching and decoding the instruction, executing the instruction, and storing the execution result in the destination identified by the instruction. More specifically, during what is referred to as a fetch stage, an instruction is fetched by the CPU 102 from the memory 104 by way of an instruction data bus 106. The CPU 102 then executes the fetched instruction.

[0003] Figure 2 illustrates an organization of the memory 104 shown in Figure 1. The memory 104 includes a cache memory 202 and an implemented memory 204. The cache memory 202 is typically divided into a scratch pad RAM (SPR) 206 and an instruction cache portion 208. In conventional program memory architectures, the SPR 206 and the instruction cache 208 are generally a fixed type architecture in that the relative sizes of the SPR 206 and the instruction cache 208 remain constant regardless of the application being executed by the CPU 102.

[0004] Applications typically have very different requirements, and for some, the benefit of a large instruction cache 208 will outweigh the need for a deterministic execution time provided by the SPR 206. Unfortunately, conventional \ memory subsystems allocate fixed portions of the available memory to the instruction cache 208 and the SPR 206 regardless of the particular requirements of the application being processed by the CPU 102. This typically results in less than optimal performance since in some instances the amount of scratchpad memory is not sufficient to accommodate those routines included in an application for which the scratchpad memory is required.

[0005] In view of the foregoing, it should be apparent that the capability of dynamically configuring the memory used to store instructions and data used by a micro-controller is desirable.

SUMMARY OF THE INVENTION

[0006] According to one aspect of the invention, a method for executing a computer program having configuration data using a CPU with a main memory and an associated cache memory is disclosed. In one embodiment of the invention, the computer program and the configuration data are loaded into the main memory. Using the configuration data, a portion of the cache memory is configured as one of a plurality of cache memory types. The computer program is then executed by fetching instructions from the configured cache memory and the main memory.

[0007] In a preferred embodiment, the configured portion of the cache memory is a scratch RAM type memory.

[0008] In another aspect of the invention, a method for dynamically configuring a cache memory is disclosed. The cache memory includes a plurality of cache lines formed of a plurality of contiguous bytes. In one embodiment, a first one of the contiguous bytes resides at a cache line memory address whereas the remaining bits of the cache line memory address form a cache line tag that is used to refer to the entire cache line. The cache memory also includes a programmable cache memory address decoder arranged to form an index used to select a particular row within the cache memory. In one embodiment, the index is based upon a selected number of bits from the cache line memory address. The method is implemented by providing configuration data associated with a particular cache memory type to the programmable cache memory address decoder and programming the cache memory address decoder based upon the configuration data to recognize a range of indexes as the particular cache memory type.

[0009] In yet another aspect of the invention, a computer system is disclosed. The computer system includes a main memory for storing a plurality of computer program instructions having associated configuration data. A central processing unit (CPU) for executing the computer program instructions, and a cache memory for storing selected ones of the computer program instructions for access by the CPU. The computer system also includes configuration circuitry for configuring a portion of the cache memory as one of a plurality of cache memory types according to the configuration data.

[0010] In still another aspect of the invention, a computer program product is disclosed. The computer program product includes a computer readable medium, and computer program instructions embedded in the computer readable medium. The computer program product also includes configuration data associated with the computer program instructions that is embedded in the computer readable medium. The configuration data being used to configure a portion of a cache memory as one of a plurality of cache memory types for facilitating execution of the computer program instructions by a computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 illustrates a conventional computing system;
Figure 2 illustrates the organization of the program memory shown in Figure 1;
Figure 3 illustrates a computing system in accordance with an embodiment of the invention;
Figure 4 illustrates a CPU bus interface in accordance with an embodiment of the invention; and
Figure 5 is a flowchart detailing a process for dynamically reconfiguring a cache memory in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0012] In the following detailed description of the present invention, numerous specific embodiments are set forth in order to provide a thorough understanding of the invention. However, as will be apparent to those skilled in the art, the present invention may be practiced without these specific details or by using alternate elements or processes. In other instances well known processes, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

[0013] Referring initially to Figure 3, an illustration of a computing system 300 in accordance with an embodiment of the invention is shown. The computing system 300 includes an instruction memory 302 connected to a processing unit 304 by way of an instruction bus 306. In the described embodiment, the instruction bus 306 is arranged to carry program data from the instruction memory 302 and corresponding to instruction addresses provided by the processing unit 304. It should be noted that for the following discussion the processing unit 304 is a dedicated central processing unit, or CPU, found in most computing systems. However, other type processing units capable of performing executable instructions are also contemplated. Such other type processing units include re-configurable devices such as for example, programmable logic devices, or PLDs, field programmable logic devices, or FPLD, found in, for example, re-configurable computer systems.

[0014] In the described embodiment, the instruction memory 302 is arranged to store executable program instructions associated with an application, or computer program, that are fetched and subsequently executed by the CPU 304. The instruction memory 302 also includes a configurable cache memory 308. The configurable cache memory 308 is arranged to use configuration data provided by the CPU 304 to allocate certain of its constituent memory locations that are used for storing selected ones of the executable instructions that form the program being performed by the CPU 304. In one implementation, a scratch pad RAM (SPR) type memory is desirable when storing those instructions which require a deterministic fetching time. In other words, the elapsed time between the signaling by the CPU 304 that an instruction is to be fetched from the memory 308 and the time it is available to the CPU 304 must be constant throughout the execution of the program. Since this constancy of memory access time can not be guaranteed in a conventionally architectured cache memory with dynamic allocation of cache lines, the static allocation of a range of memory addresses within the memory 308 as the SPR type memory is a very desirable attribute of the invention. Even more so, is the capability of dynamically re-configuring the memory 308 based upon the requirements of the particular program being performed by the CPU 304.

[0015] In one embodiment of the invention, a system bus 310 connects the CPU 304 to a system bus interface 312 included in the memory 308. The system bus interface 312 is also connected to a cache memory array 314 also included in the configurable cache memory 308. In one embodiment, the system bus interface 312 uses the configuration data to configure the cache memory array 314 into appropriate cache memory types. The instruction memory 302 and the configurable cache memory 308 are connected to the instruction bus 306 by way of a program memory/CPU interface 316.

[0016] In the described embodiment, the CPU 304 is also connected to a data memory 320 by way of the system bus 310. The system bus 310 is, in turn, connected to a system bus interface 322 included in the data memory 320. In the described embodiment, a data bus 324 connecting the CPU 304 to a data memory/CPU interface 326 included in the data memory 320 is arranged to carry data from the CPU 304.

[0017] In one embodiment of the invention, the configurable cache memory array 314 stores instructions from the instruction memory 302 in what is referred to as cache lines. A cache line is formed of a plurality of contiguous bytes that are typically aligned such that the first of the contiguous bytes resides at an address having a certain number of low order bits set to zero. The certain number of low order bits is sufficient to uniquely identify each byte in the cache line. The remaining bits of the address form a tag that is used to refer to the entire cache line.

[0018] Cache memories are typically organized into an associative structure (also referred to as "set associative"). In an associative structure, the cache lines are accessed as a two dimensional array having rows and columns. When a cache memory is searched for bytes residing at an address, a number of bits from the address are used as an index into the cache. The index selects a particular row within the two dimensional array, and therefore, the number of address bits required for the index is determined by the number of rows configured into the cache. The act of selecting a row via an index is referred to as "indexing". The addresses associated with bytes stored in the multiple cache lines of a row are examined to determine if any of the addresses stored in the row match the requested address. If a match is found, the access is said to be a "hit" and the cache provides the associated bytes. If, however, a miss is detected, the bytes are transferred from the instruction memory 302 into the cache memory 308. It should be noted that the bytes could also come from a system bus if a cacheable memory is coupled to the bus as shown in Figure 3.

[0019] During operation, computer program instructions and associated configuration data are stored in the instruction memory 302. Prior to execution of the computer program instructions, the configuration data is retrieved by the CPU 304 and conveyed by way of the CPU bus 310 to the CPU bus interface 312. The CPU bus interface 312 uses the configuration data to configure the cache memory array 314 into appropriate cache memory types. In one embodiment of the invention, the configuring includes allocating certain of the cache memory address space represented by, for example, a range of cache memory addresses, to particular cache memory types. By way of example, a range of cache memory addresses from {0100} to {0400} would represent an SPR cache memory type region in which those computer program instructions requiring a deterministic fetch time would be stored. In this way, each cache memory type has associated with it a corresponding index that is used to select those rows located within the appropriate cache memory type region.

[0020] Subsequent address requests from, for example the CPU 304 in the form of a CPU memory request, are decoded and compared to the allocated range of memory addresses which delineate the configured cache memory array 314. When the CPU memory request, for example, is found to be located within this range, the corresponding memory location within the cache memory type is made accessible to the requesting device.

[0021] By using the configurable cache memory array 314 in conjunction with the configuration data, instructions requiring, for example, a deterministic fetching and execution time can be stored in, for example, a dynamically allocated SPR type memory portion of the cache memory array 314. By way of example; if the cache memory array 314 has a total memory storage capability of 2K (2048) bytes arranged in appropriate columns and rows, 1K bytes can be apportioned as SPR type memory as determined by the CPU 304 based upon the application being processed. In this case, any instruction stored in the SPR region (as determined by the allocated memory addresses associated with the SPR memory region) of the cache memory array 314 will have a deterministic fetch time. On the other hand, if the cache memory array 314 is configured to have no SPR memory region, then the entire 2K bytes are treated as a conventional cache memory. Conversely, the cache memory array 314 can use the configuration data can be used to apportion the entire cache memory array 314 (2K bytes) or any portion thereof as SPR memory.

[0022] Referring now to Figure 4, a CPU bus interface 400 in accordance with an embodiment of the invention is shown. It should be noted that the CPU bus interface 400 is one embodiment of the CPU bus interface 312 shown in Figure 3. In one embodiment of the invention, the CPU bus interface 400 couples the CPU bus 310 to the cache memory array 314. During the initial configuration of the cache memory array 314, the CPU bus 310 provides configuration data to a configuration data register 402. The configuration data is then made available to a programmable cache address decoder 404. The programmable cache address decoder 404 is, in turn, arranged to provide a cache memory address access signal based upon a CPU address generated by the CPU 304 and the configuration data. In one embodiment of the invention, the CPU bus interface 400 includes a CPU address segment decoder 406 connected to the CPU bus 310 and the programmable cache address decoder 404. The CPU address segment decoder 402 is also connected to a cache memory range comparator 408. In the described embodiment, the cache memory range comparator 408 is used to determine if a memory address request received by the CPU interface 400 is located within the appropriate region of the memory space of the cache memory array 314 allocated to a particular cache memory type. In the case where the received memory address request is within the appropriate cache memory space, the memory range comparator 408 provides a valid memory decode signal to the programmable cache memory decoder 404 by way of an input line 410. In the case, however, where the received memory address request is not within the range of cache memory addresses allocated to the particular cache memory type, in one embodiment, an error message is generated.

[0023] It should be noted that in one embodiment, the programmable cache memory address decoder 404 includes a selector 412 connected to the CPU bus 310 arranged to receive the memory address request and generate an appropriate index corresponding to the particular cache memory type. In the described embodiment, the selector 412 uses configuration data stored in the configuration data register 402 and the valid memory range signal to generate the index. By way of example, if a selected range of cache memory addresses of the cache memory array 314 have been allocated to the particular cache memory type cache memory array 314, the selector 314 will provide the appropriate index for those instructions stored therein.

[0024] During a fetch operation, the CPU bus 310 conveys a memory request generated by the CPU 304 to the CPU bus interface 400. The CPU bus interface 400 uses the configuration data associated with the memory location requested by the received CPU memory request to ascertain if the memory location corresponding to the received CPU memory request is located within the range of memory addresses allocated for particular cache memory type. If it is found to be within the proper range of memory addresses, an appropriate cache memory address request signal is generated by the programmable cache memory address decoder 404.

[0025] Using the example from above of the 4 GB memory, the corresponding CPU memory request is a 32-bit data word. In this example, the 4 most significant bits (MSB) would be used to indicate if the associated CPU memory request is located within the appropriate one of sixteen memory segments corresponding to the SPR segment. If the memory range comparator unit 408 determines that the CPU memory request is not within the range of memory addresses allocated to an SPR memory, for example, then an appropriate error signal is generated. In that case, the CPU bus interface 400 can pass control back to the CPU 304 for further instructions or execution can proceed, for example, by fetching the required instruction from the appropriate memory segment indicated by the CPU memory request segment.

[0026] When the memory range comparator 408 determines that the received CPU memory request is located within the appropriate memory space allocated for the SPR memory, a valid memory address decode signal is generated.

[0027] Assuming for this example only that the cache memory 308 is a 2 way set associative cache memory having 1K bytes of memory arranged as 32 cache lines of 256 bits per cache line where each cache line includes 4 data words of 64 bits each. In this arrangement, one half of the cache memory (i.e., 16 cache lines) are allocated for SPR type memory. In this case, the cache memory address signal used to access a particular memory location in the cache memory array 314 includes at least a 4-bit index used to point to a particular cache line. The cache memory address signal also includes an access bit used to identify the portion of the cache memory 308 (i.e.; either the SPR type memory or the conventional cache memory) in which the indexed cache line resides.

[0028] By way of example, for the 1K cache memory divided into 32 cache lines, the corresponding cache memory address would be a 5 bit data word having the {MSB}th as the access bit and the {MSB-1 ... MSB-4}th bits as the cache line index. In this case, the programmable cache memory decoder 404 would be configured by the configuration data to recognize a MSB = 1 to indicate, for example, that the cache line associated with the remaining {MSB-1 ... MSB-4} bits as residing in the conventional cache memory region of the cache memory 308. Alternatively, a MSB = 0 would indicate that the cache line would reside in the SPR type memory region of the cache memory 314.

[0029] In other words, with a 1K bytes 2 way set associative cache system, with each cache line made of 256 bits, the cache system is arranged as 256 (bits/line) x 32 (lines) for a total of 1K bytes. These 32 lines are equally divided into 2 sets (16 lines each). In order to address a cache line, 4 bits are required to address a particular line in a set and an extra bit to choose between sets.

[0030] Figure 5 is a possible process 500 for executing a program using a dynamically reconfigurable cache memory in accordance with an embodiment of the invention. The process 500 will be discussed with reference to Figures 3 and 4. The computer program instructions and associated configuration data used to configure the cache memory are stored in a program memory (502). The configuration data is then provided to the configurable cache memory (504). The configurable cache memory then designates selected ranges of memory locations as particular cache memory types based upon the configuration data (506). Next, a memory request corresponding to the memory location at which the instruction to be fetched is stored, is generated in furtherance of the program execution (508). Next, it is determined if the memory request is located within the memory space allocated by the configuration data to be within the particular cache memory type (510). If it is determined that the received CPU memory request is not within the particular memory space, an error signal is generated (512). Otherwise, the appropriate cache memory address is generated (514) based upon the received CPU memory request and the configuration data and passed to a programmable address decoder. The programmable address decoder then accesses the appropriate memory location within the configurable cache memory (516) at which the instruction to be fetched is stored.

[0031] Although only a few embodiments of the present invention have been described in detail, it should be understood that the present invention can be embodied in many other specific forms without departing from the spirit or scope of the invention. Particularly, although the invention has been described primarily in the context of computer systems having processor and memory subsystems, the advantages including improved CPU performance due to the ability to dynamically reconfigure the instruction cache can be applied to any system where dynamically configurable memory is desirous. Such systems include reconfigurable computing systems having no CPU but configurable logic devices (such as PLDs) that act as on the fly controllers.

[0032] Additionally, the characteristics of the invention can be varied in accordance with the needs of a particular system. Therefore, the present examples are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.

Claims

1. A method for executing a computer program using a CPU (304) having a main memory and a cache memory associated therewith, the computer program having configuration data associated therewith, comprising:

   loading the computer program into the main memory (302);
   configuring a portion of the cache memory (314) as one of a plurality of cache memory types in accordance with the configuration data associated with the computer program; and
   fetching instructions from the cache memory (314) and the main memory for execution of the computer program.

2. A method as recited in claim 1, wherein the configuring comprises:

   programming a programmable cache memory address decoder (404) connected to the cache memory (314) based upon the configuration data to recognize a range of memory addresses in the cache memory (314) as being a particular cache memory type.

3. A method as recited in claim 2, wherein the fetching comprises:

   decoding a segment of a received memory address request by an address segment decoder (406), wherein the memory address request identifies the memory address in the cache memory (314) associated with the instruction to be fetched; and
   storing configuration data associated with the received memory address request in a configuration register (402) that is connected to the programmable cache memory address decoder (404); and
   determining by a memory range comparator (408) connected to the configuration register (402) and the segment decoder (406) if the received memory address request is located within the range of memory locations recognized as being the particular cache memory type.

4. A method as recited in claim 3, wherein the determining is based upon the configuration data and the decoded memory address request segment.

5. The method as recited in claim 4, wherein the determining further comprises:

   generating a valid decode signal by the memory range comparator (408) when the received address is determined to be located within the range of memory addresses recognized as being the particular cache memory type, wherein the valid decode signal is used by the programmable decoder (404) in decoding the received memory address request.

6. A method as recited in claim 5, wherein the fetching comprises:

   decoding the received memory address request by the programmable cache memory address decoder (404) using the valid decode signal and the configuration data by the to form a corresponding cache memory address request; and
   accessing the location in the cache memory (314) corresponding decoded received memory address request at which the instruction to be fetched is stored.

7. The method as recited in claim 6, wherein the particular cache memory type is a scratch pad RAM.

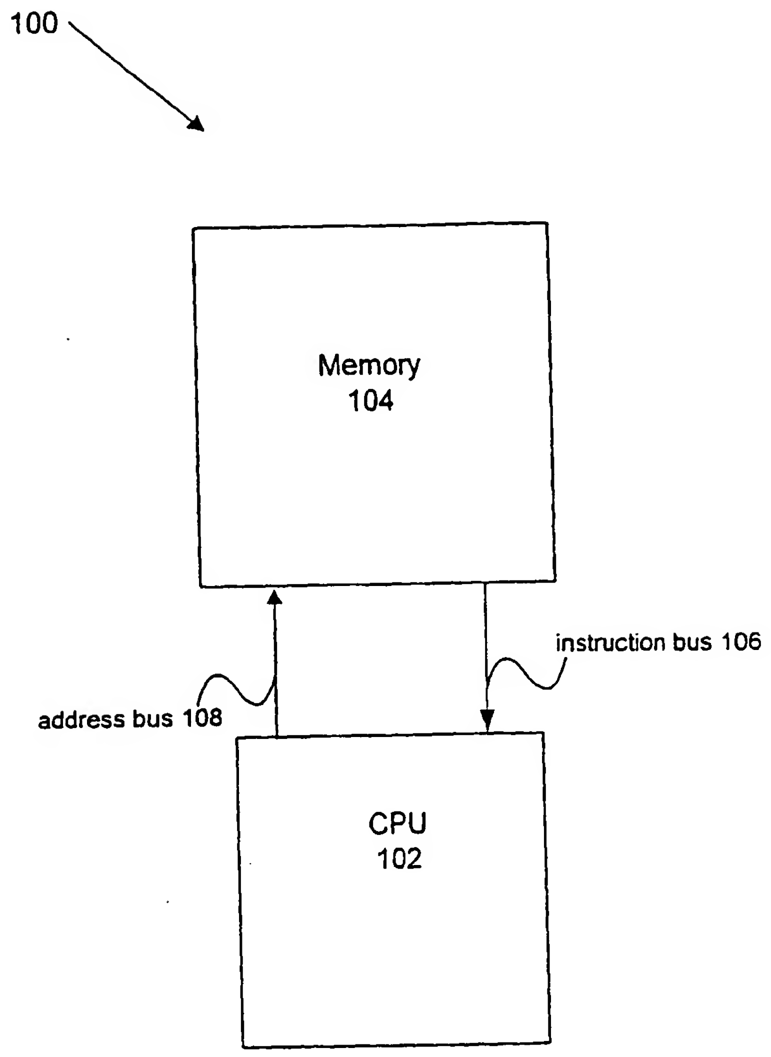


FIGURE 1

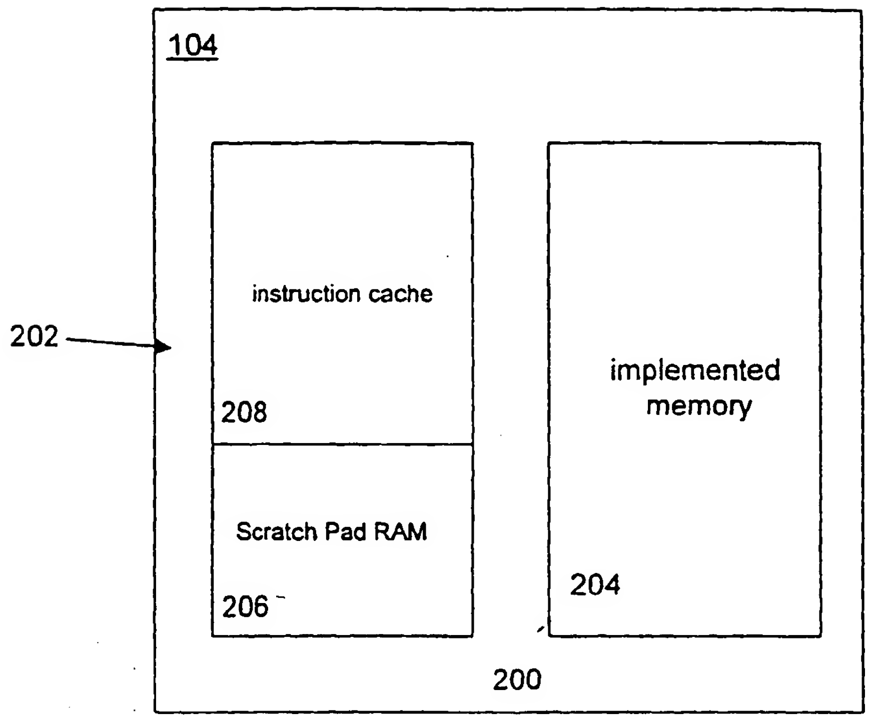

104
instruction cache
208
Scratch Pad RAM
206
implemented memory
204
202
200

FIGURE 2

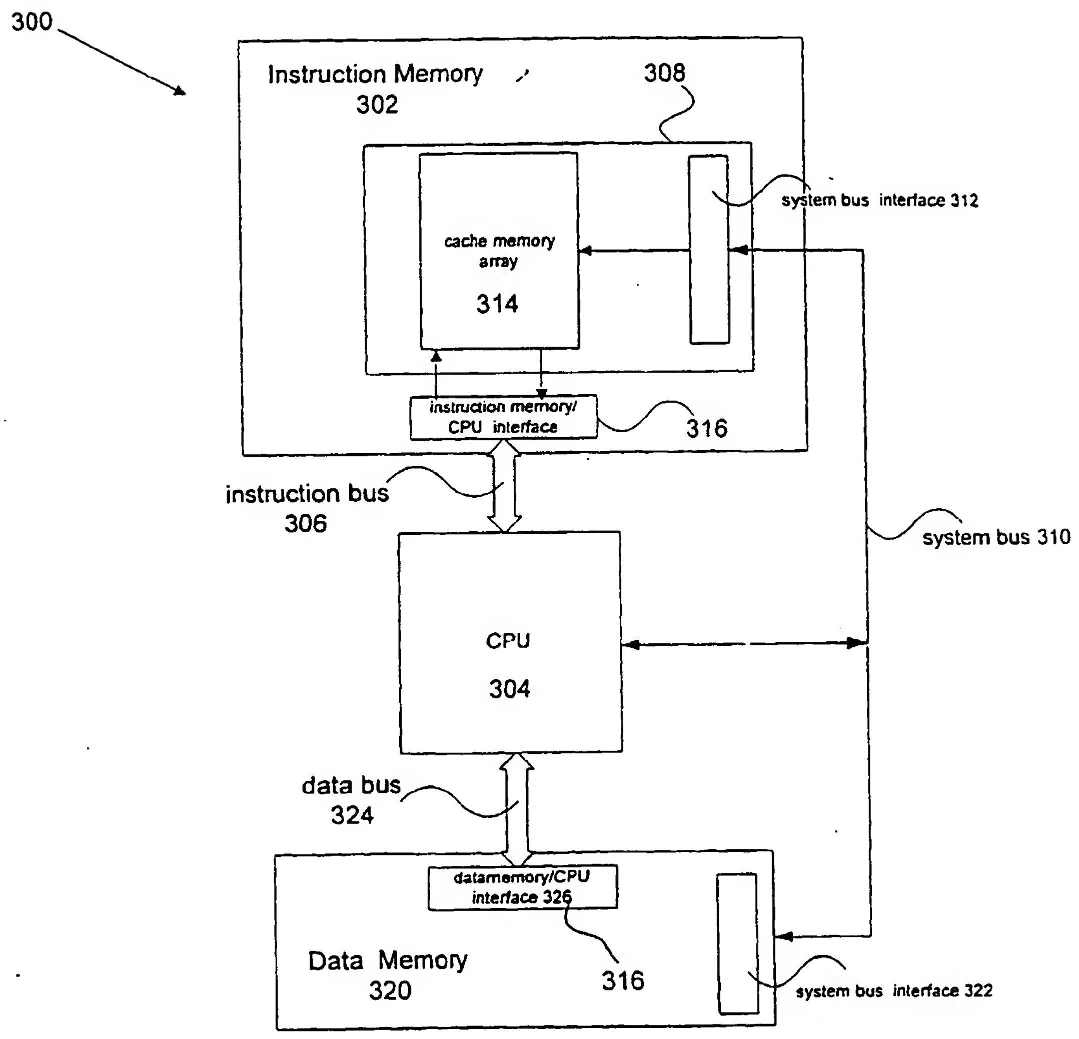
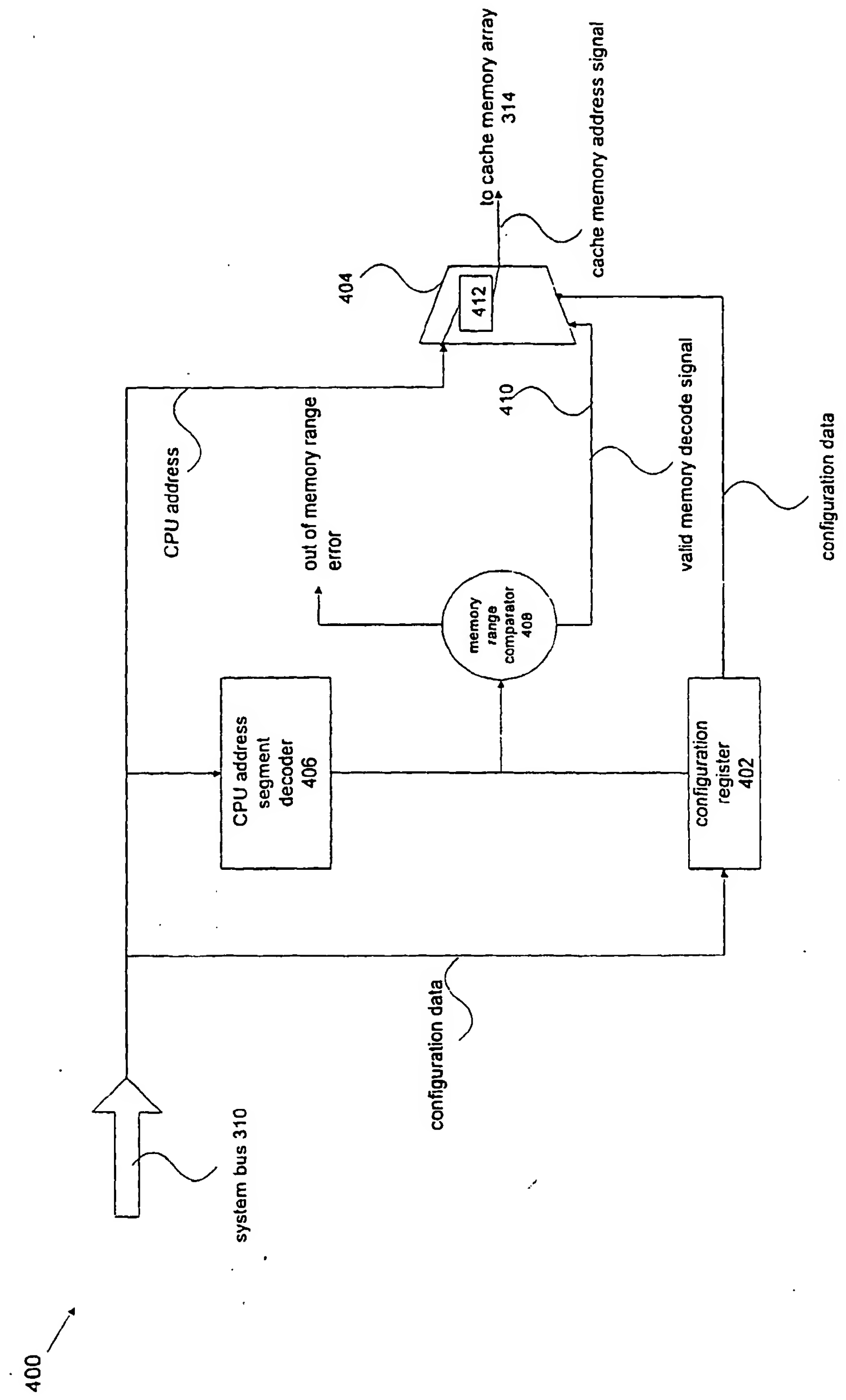

300
Instruction Memory
302
308
system bus interface 312
cache memory array
314
instruction memory/ CPU interface
316
instruction bus
306
system bus 310
CPU
304
data bus
324
datamemory/CPU interface 326
Data Memory
320
316
system bus interface 322

Figure 3



400
system bus 310
configuration data
CPU address
CPU address segment decoder 406
configuration register 402
memory range comparator 408
out of memory range error
valid memory decode signal
configuration data
410
404
412
cache memory address signal
to cache memory array 314

FIGURE 4

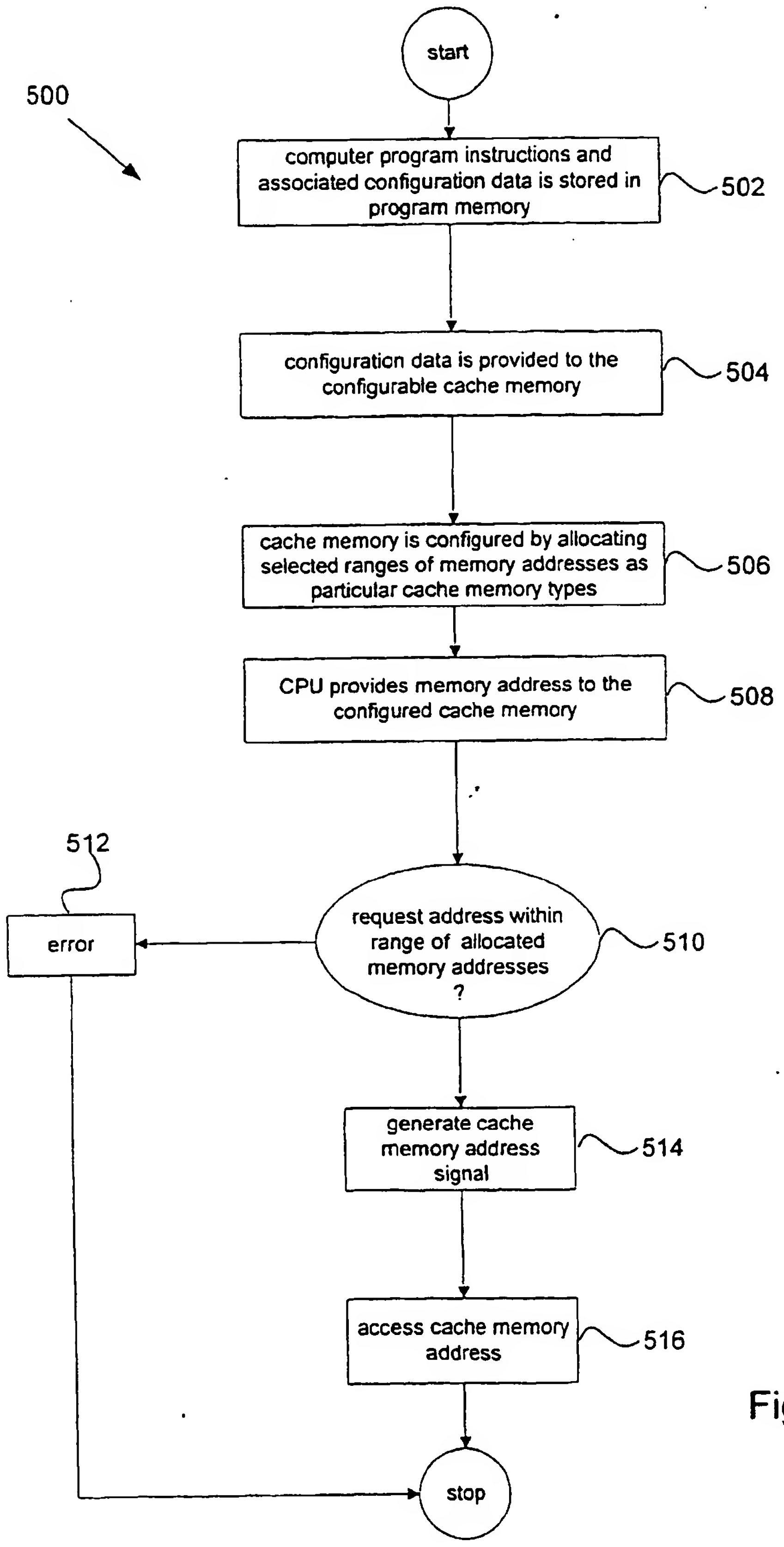

500
start
computer program instructions and associated configuration data is stored in program memory
502
configuration data is provided to the configurable cache memory
504
cache memory is configured by allocating selected ranges of memory addresses as particular cache memory types
506
CPU provides memory address to the configured cache memory
508
512
error
request address within range of allocated memory addresses ?
510
generate cache memory address signal
514
access cache memory address
516
stop

Figure 5

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 045 307 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
08.11.2000 Bulletin 2000/45

(43) Date of publication A2:
18.10.2000 Bulletin 2000/42

(21) Application number: **00106930.1**

(22) Date of filing: **31.03.2000**

(51) Int. Cl.^{7}: **G06F 12/08**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **16.04.1999 US 293278**

(71) Applicant:
**Infineon Technologies North America Corp.
San Jose, CA 95112-6000 (US)**

(72) Inventors:
- **Gautho, Manuel O.
San Jose, CA 95126 (US)**
- **Afsar, Muhammad
San Diego, CA 92129 (US)**

(74) Representative:
**Patentanwälte
Westphal, Mussgnug & Partner
Waldstrasse 33
78048 Villingen-Schwenningen (DE)**

(54) Dynamic reconfiguration of a micro-controller cache memory

(57) A configurable cache memory (314) for improving the performance of a central processing unit (CPU) (304) in the execution of a program is described. The configurable cache memory (314) provides scratch pad RAM based upon the particular requirements of the program being executed by the CPU (304). The CPU (304) provides configuration data based upon the program that is used by the configurable cache memory (314) in reallocating its memory space accordingly.

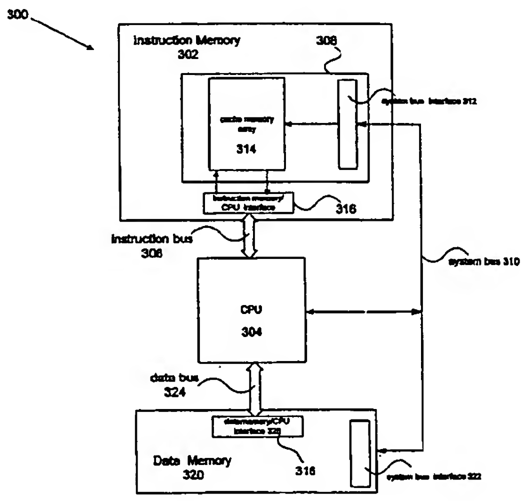


Figure 3

EP 1 045 307 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 10 6930

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	GB 2 284 911 A (PLESSEY SEMICONDUCTORS LTD) 21 June 1995 (1995-06-21)	1,2	G06F12/08
Y	* page 3, line 2 - page 4, line 7; figures 1,2 *	3-6	
X	US 4 580 240 A (WATANABE TADASHI) 1 April 1986 (1986-04-01)	1	
A	* column 3, line 21 - column 4, line 66 * * column 8, line 61 - column 9, line 17; figures 1-3 *	2-6	
Y	US 3 742 458 A (INOUE T ET AL) 26 June 1973 (1973-06-26) * abstract; claim 1; figure 3 *	3-6	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	19 September 2000	Ledrut, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 00 10 6930

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

19-09-2000

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
GB 2284911 A	21-06-1995	NONE	
US 4580240 A	01-04-1986	JP 1372069 C	07-04-1987
		JP 58102381 A	17-06-1983
		JP 61036667 B	19-08-1986
		FR 2519460 A	08-07-1983
US 3742458 A	26-06-1973	JP 49030578 B	14-08-1974
		AU 3409271 A	05-04-1973

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82